Google

HotChips 2019 Tutorial
Cloud TPU:
Codesigning Architecture and Infrastructure

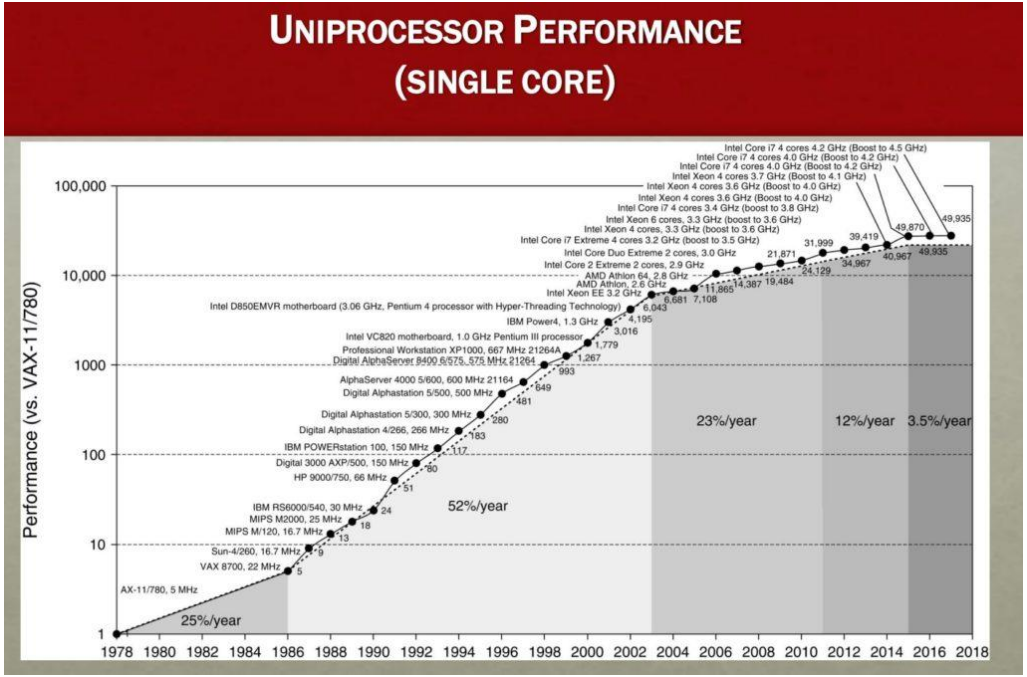
Clifford Chao
Brennan Saeta

How to Codesign: Leverage expertise across disciplines and stakeholders for optimal results.

But First...Why Codesign?

Gordon Moore: “No exponential is forever”

- Past: take general-purpose compute and shrink technology to improve performance
- Current reality: many devices not scaling well anymore. Running out of tricks!
- Solution: many gains left to be had through specializing compute to specific applications. ***Codesign!***



Codesign: TPU Architecture

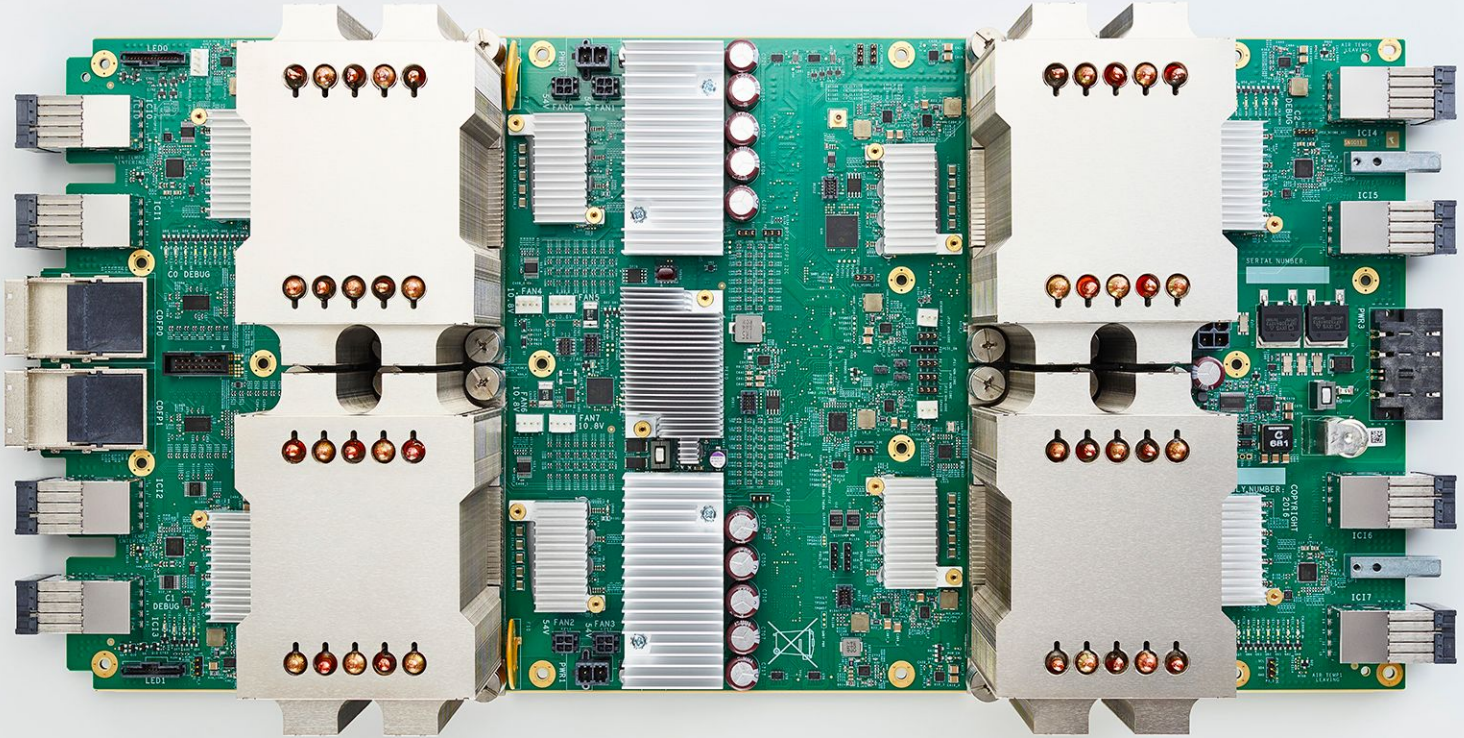
Brief Historical Perspective: TPU v1 (Google I/O 2016)

Google internally deployed in 2015 (not Cloud).

Inference workloads, including AlphaGo (Go matches again Lee Sedol).

ISCA paper in 2017, describing technical details.

Cloud TPU v2 Board (Google I/O 2017)



- For inference and training workloads
- 180 teraflops of computation, 64 GB of HBM memory, 2400 GB/s memory BW

Cloud TPU v2 Board

TPU Codesign

- **ML research**: computational requirements for cutting-edge models.
- **Systems**: power delivery, board space.
- **Data Center**: cooling, buildability.

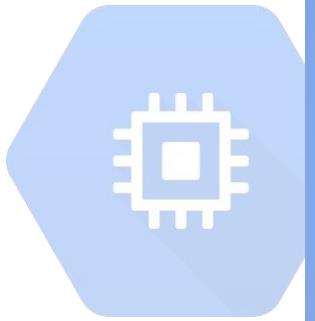
- 180 teraflops of computation, 64 GB of HBM memory, 2400 GB/s memory BW
- Architecture co-designed with ML research, systems and data center teams.

Cloud TPU v2 Host Connection

32 lane PCIe gen3

Host Server

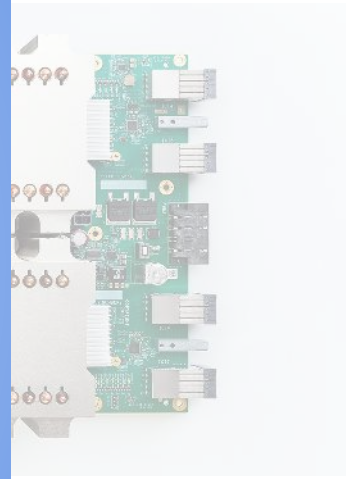
Cloud TPU v2 Host Connection



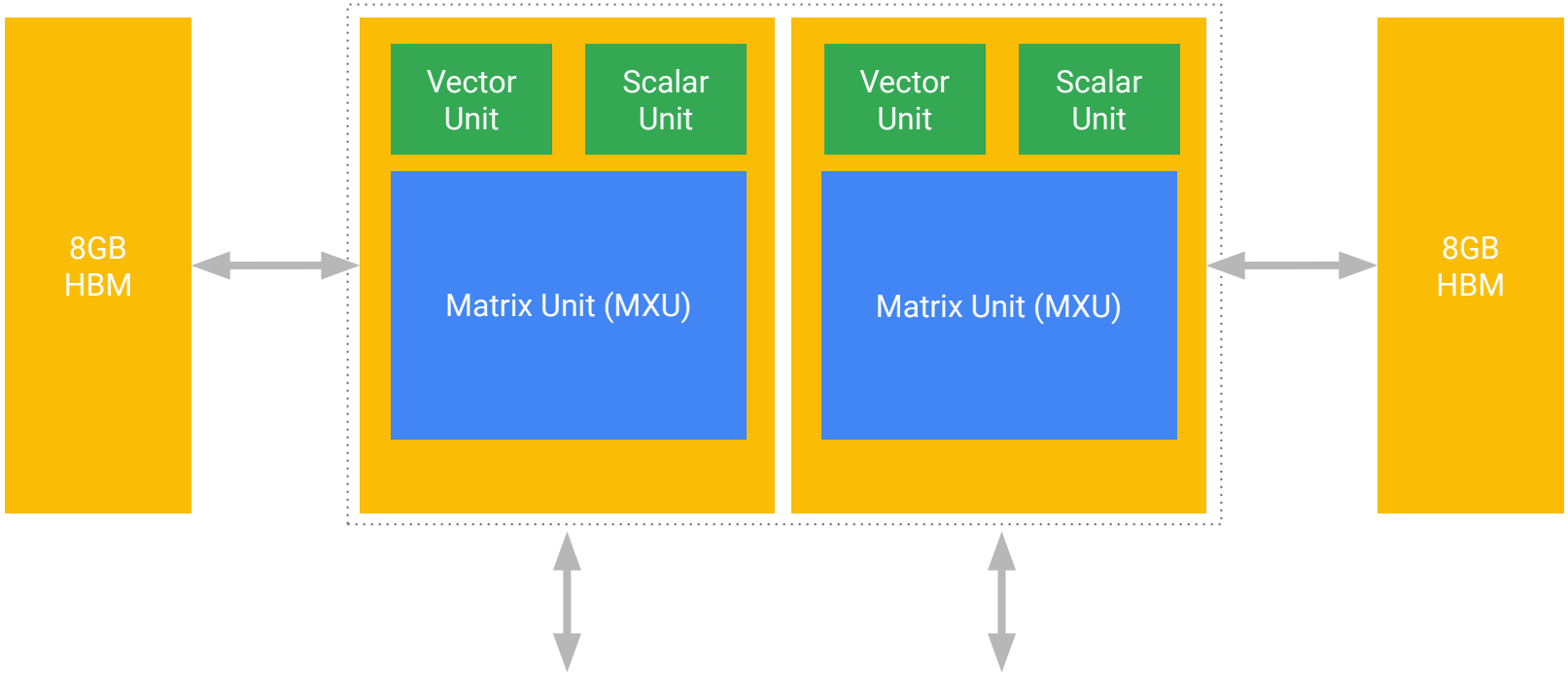
Host Serv

TPU Codesign

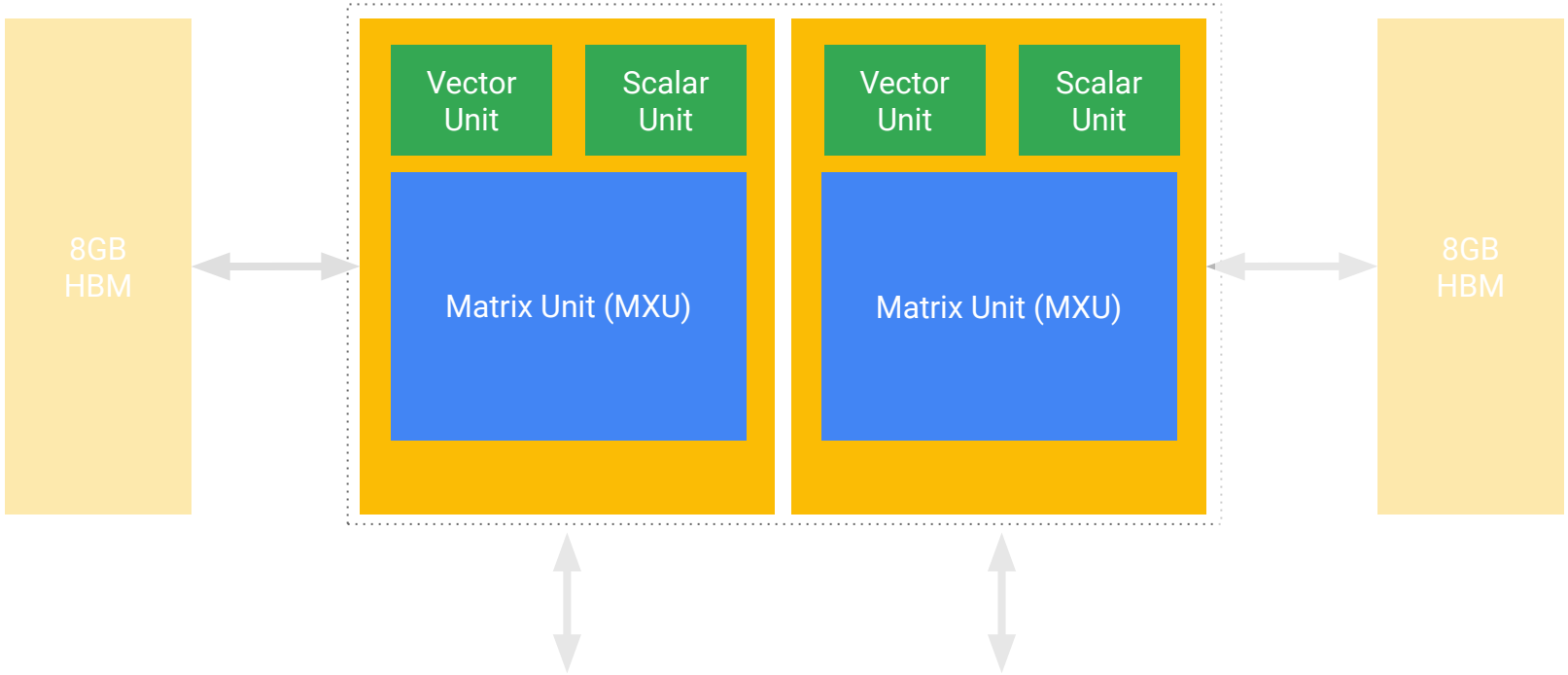
- **ML research**: input/output data feed rates.
- **Systems**: board layout.
- **Data Center**: wiring and serviceability.



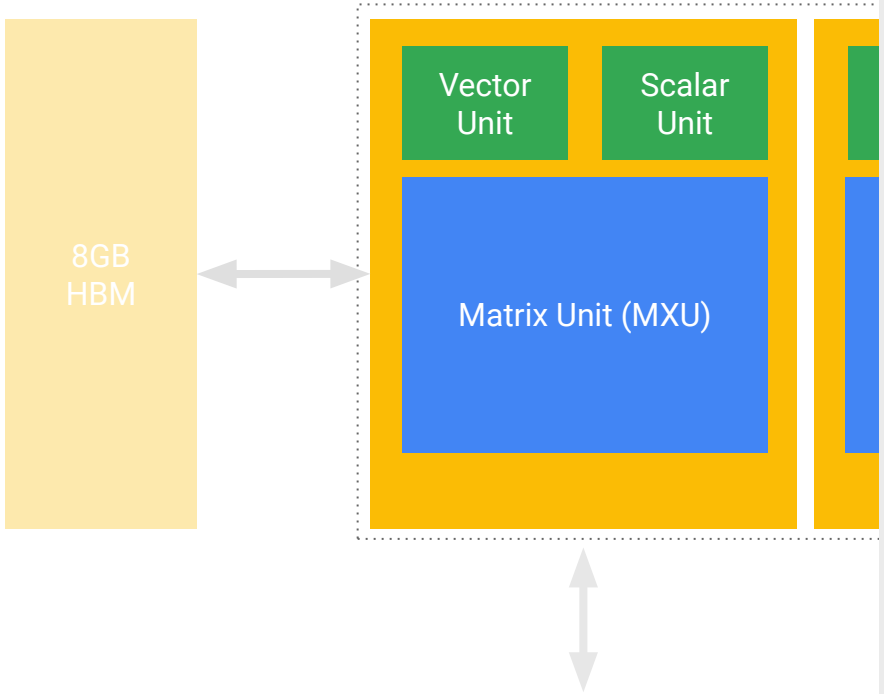
Cloud TPU v2 Chip Architecture



Cloud TPU v2 Chip Architecture



Cloud TPU v2 Chip Architec



22.5 TFLOPS per core

- 2 cores per chip
- 4 chips per board
- Scalar Unit
- Vector Unit
- Matrix Unit
- Mostly float32

Cloud TPU v2 Chip Architec

8GB
HBM

per core

r chip

r 180

ud TPU

t

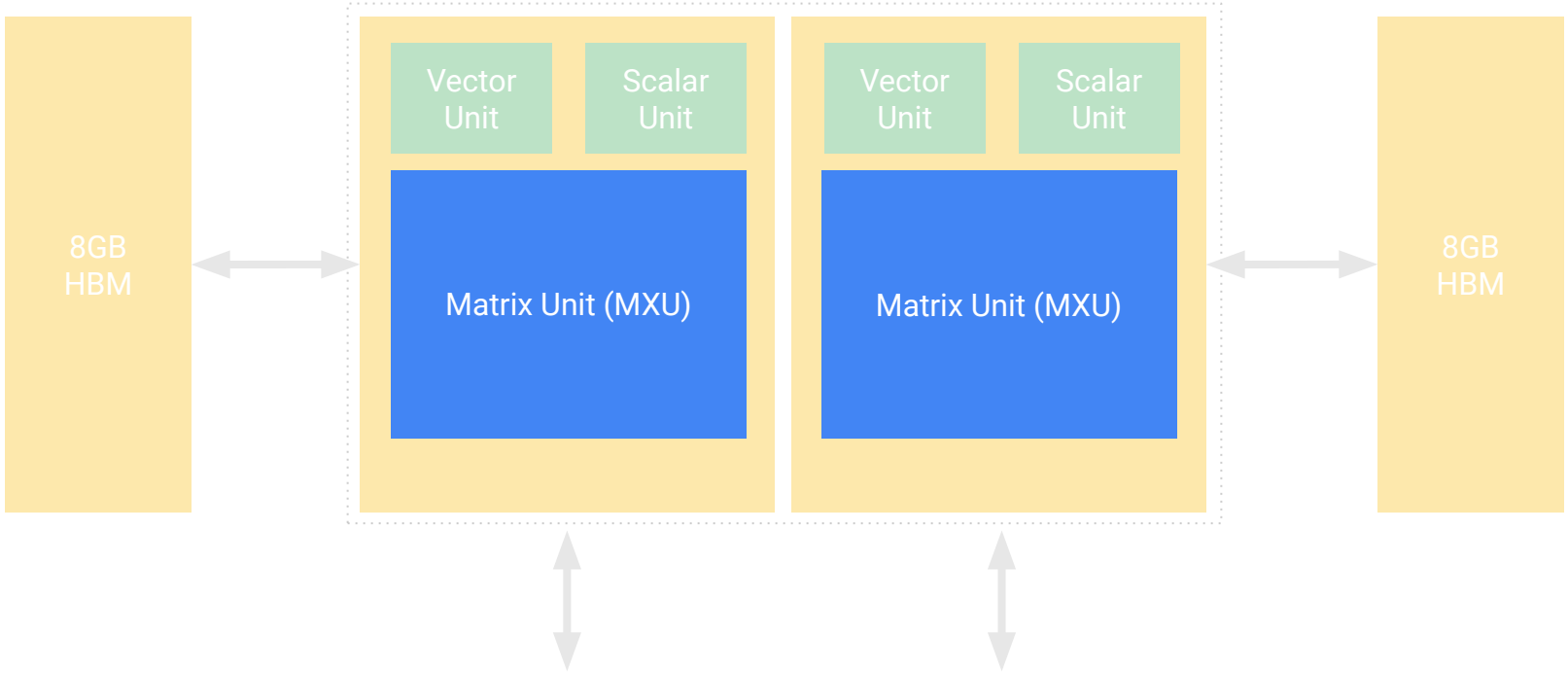
t

at32

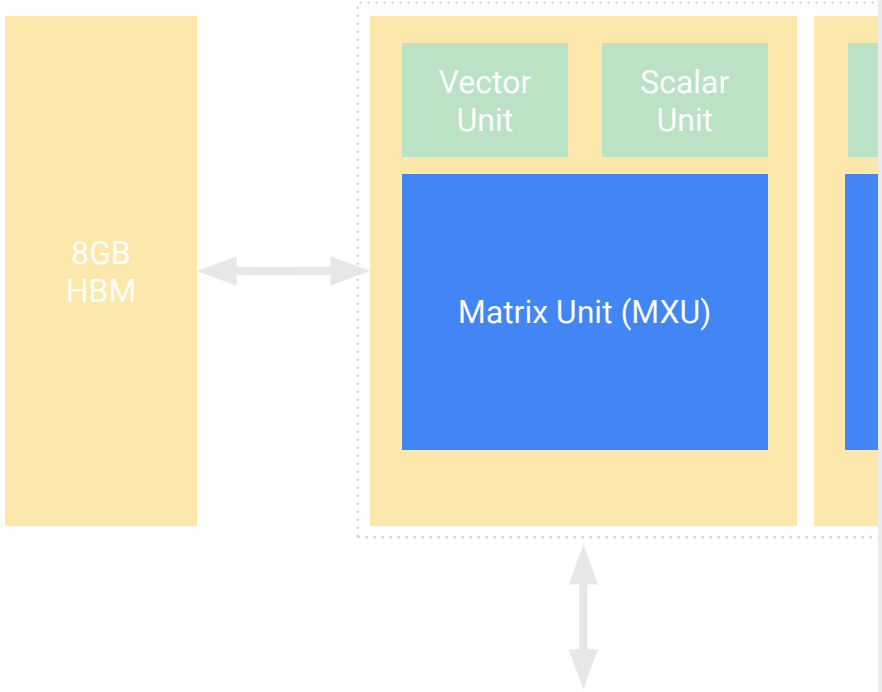
TPU Codesign

- **ML research**: types of operations to accelerate.
- **Software**: proper flexibility and programmability.
- **Data center**: performance metrics.

Cloud TPU v2 Chip Architecture



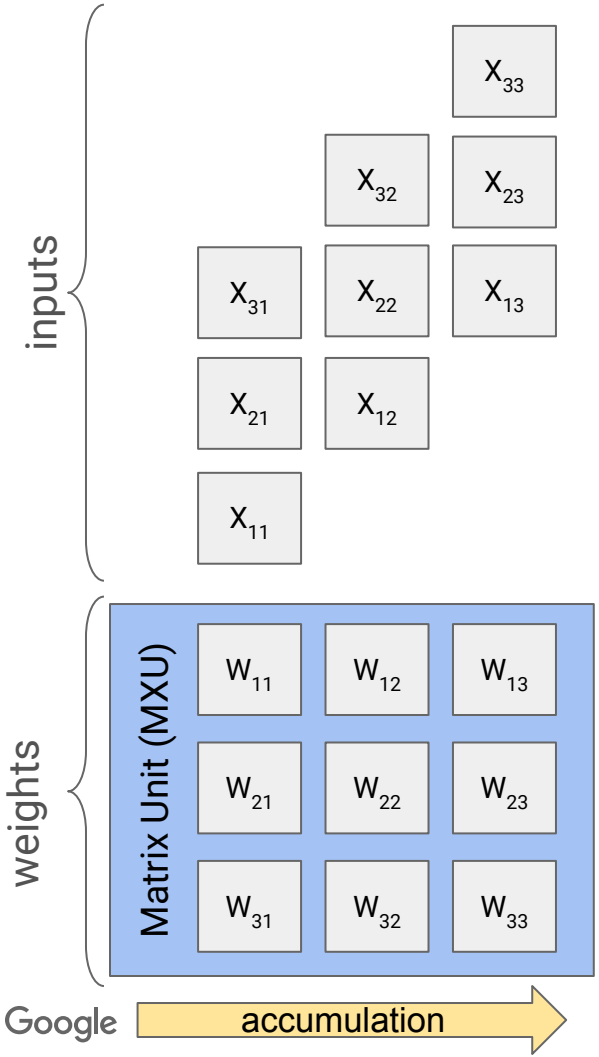
Cloud TPU v2 Chip Architec



Matrix Unit (MXU)

- 128 x 128 systolic array
- float32 accumulate
- bfloat16 multiplies

Matrix Unit Systolic Array



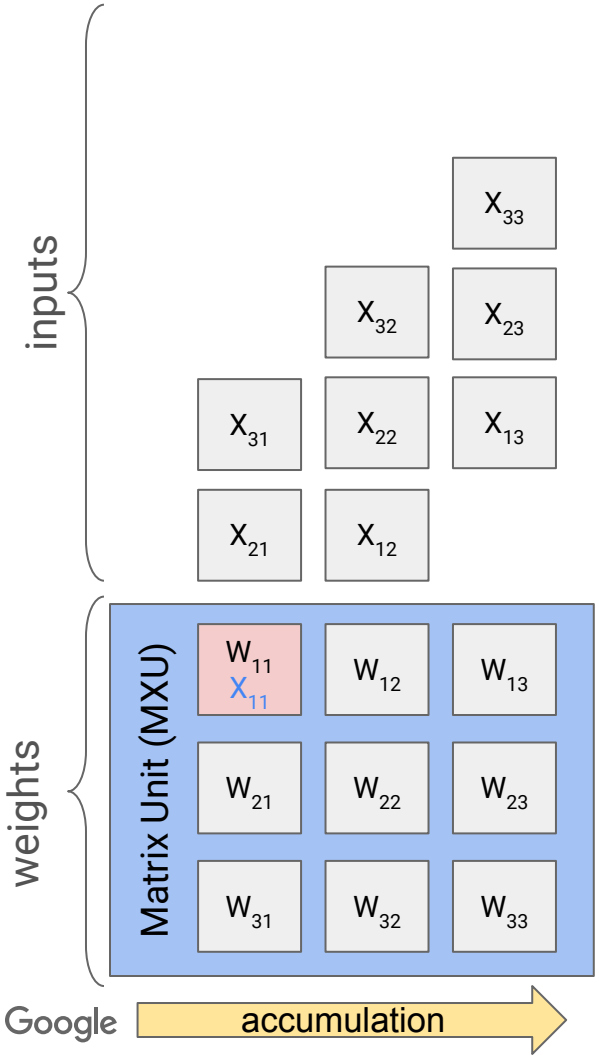
Computing Y = WX

W = 3x3 matrix
X = 3-elem vector
Y = 3-elem vector

Matrix Unit Systolic Array

Computing Y = WX

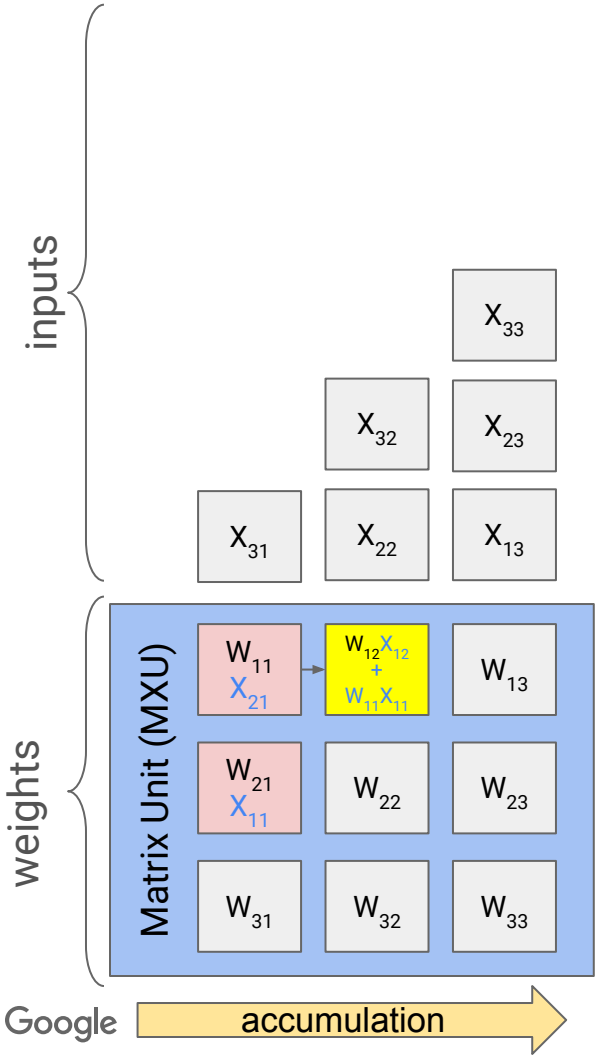
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

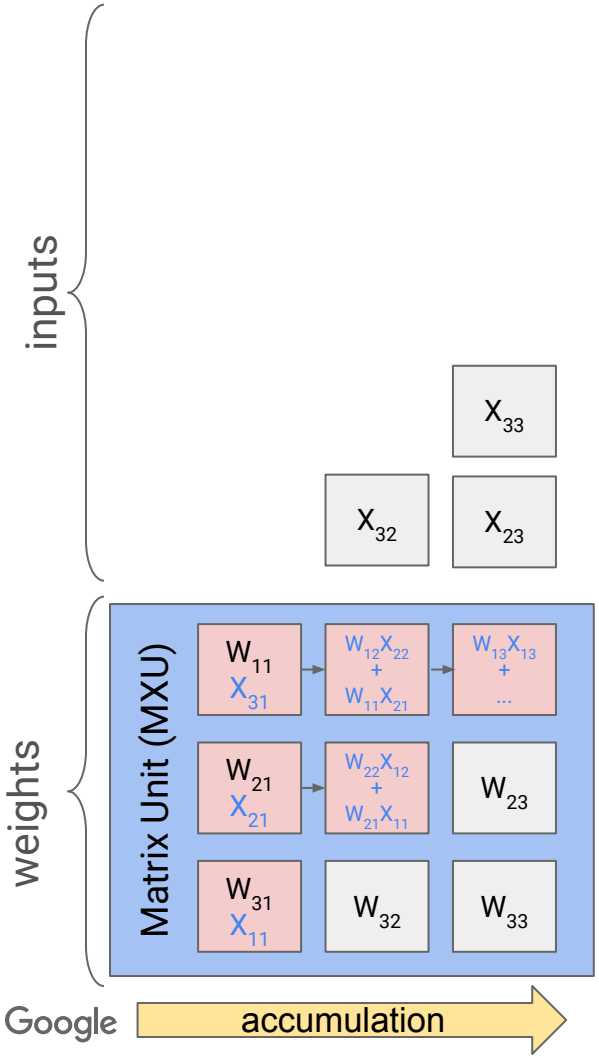
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

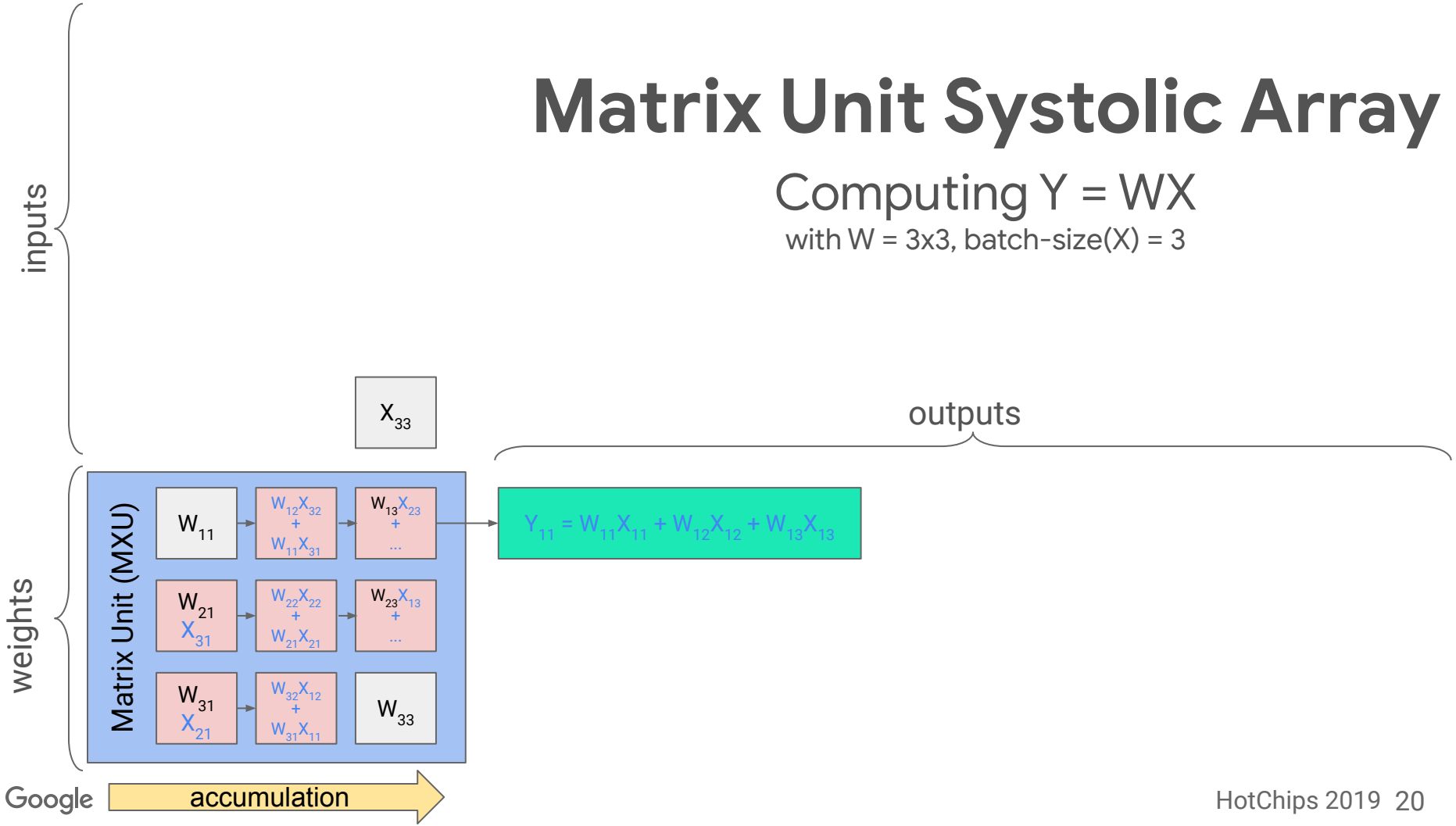
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

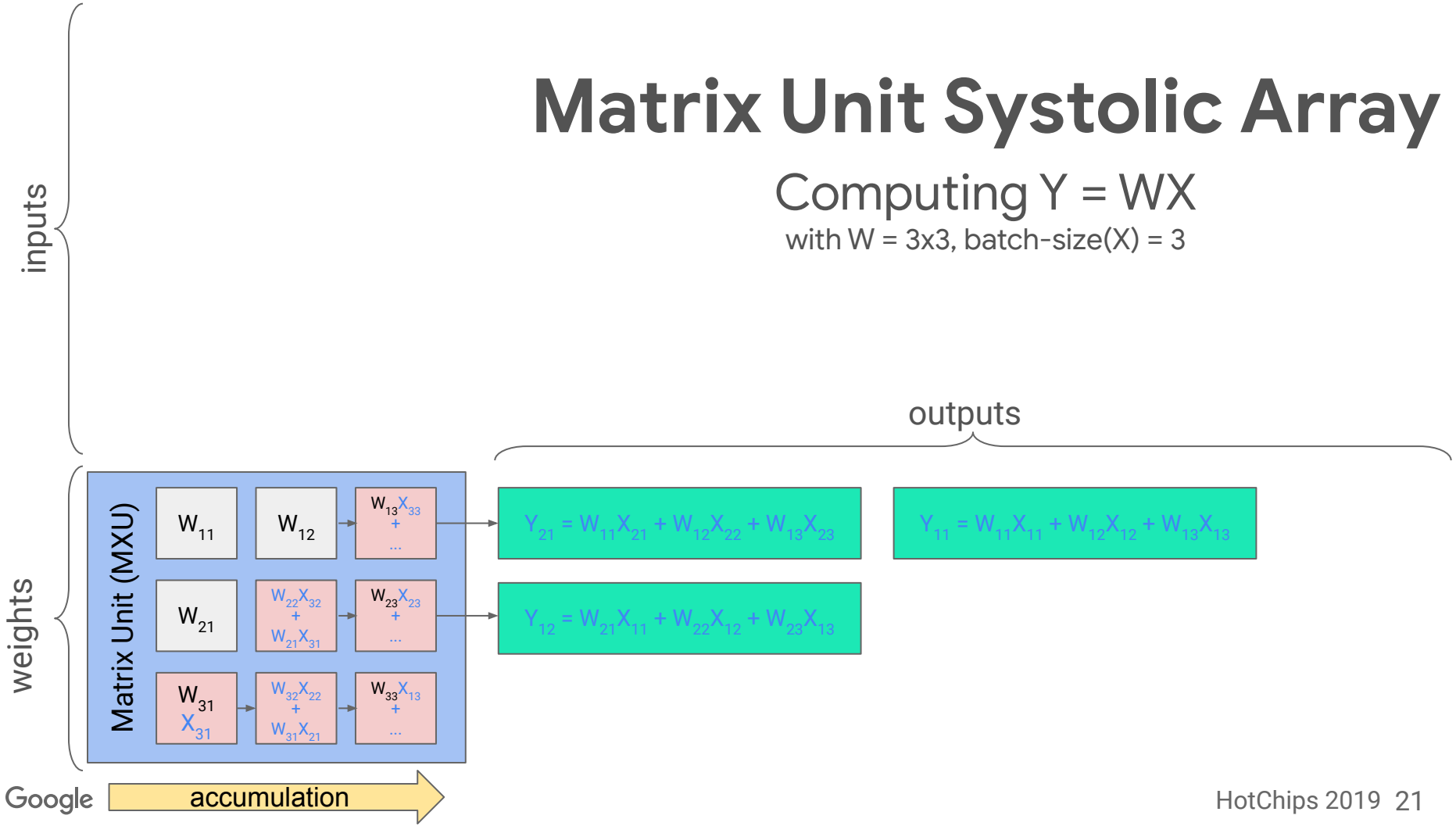
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

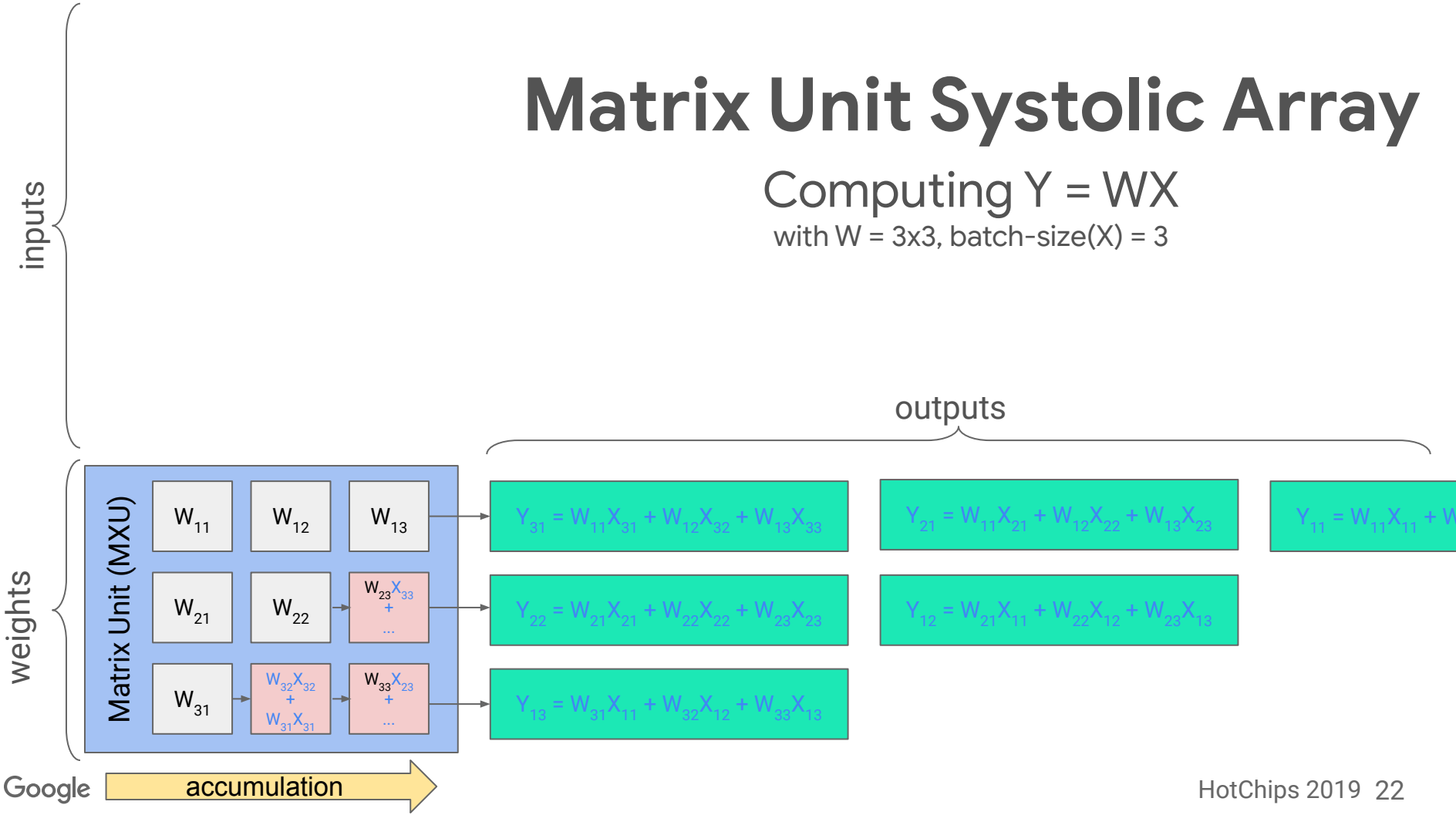
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

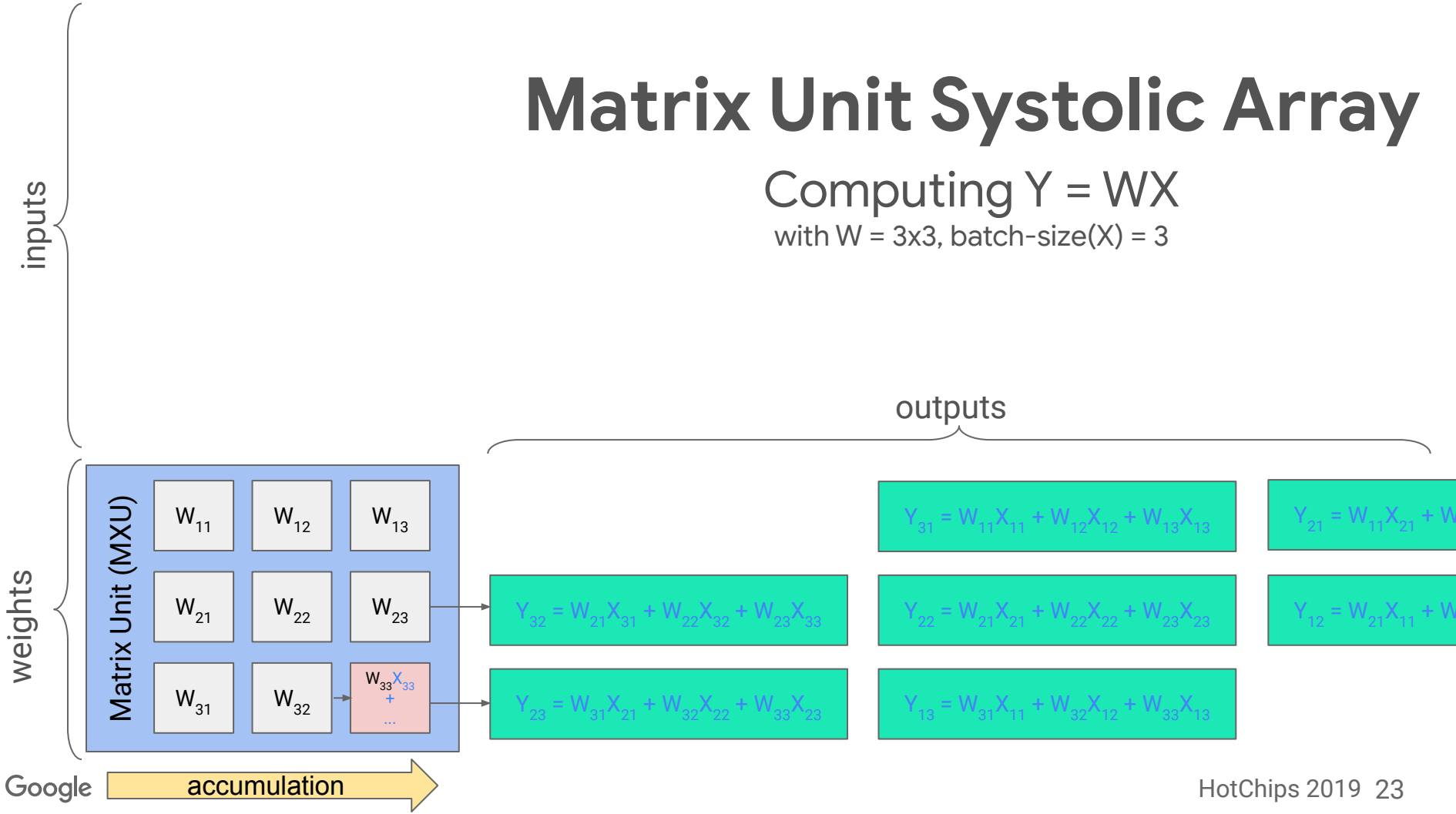
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

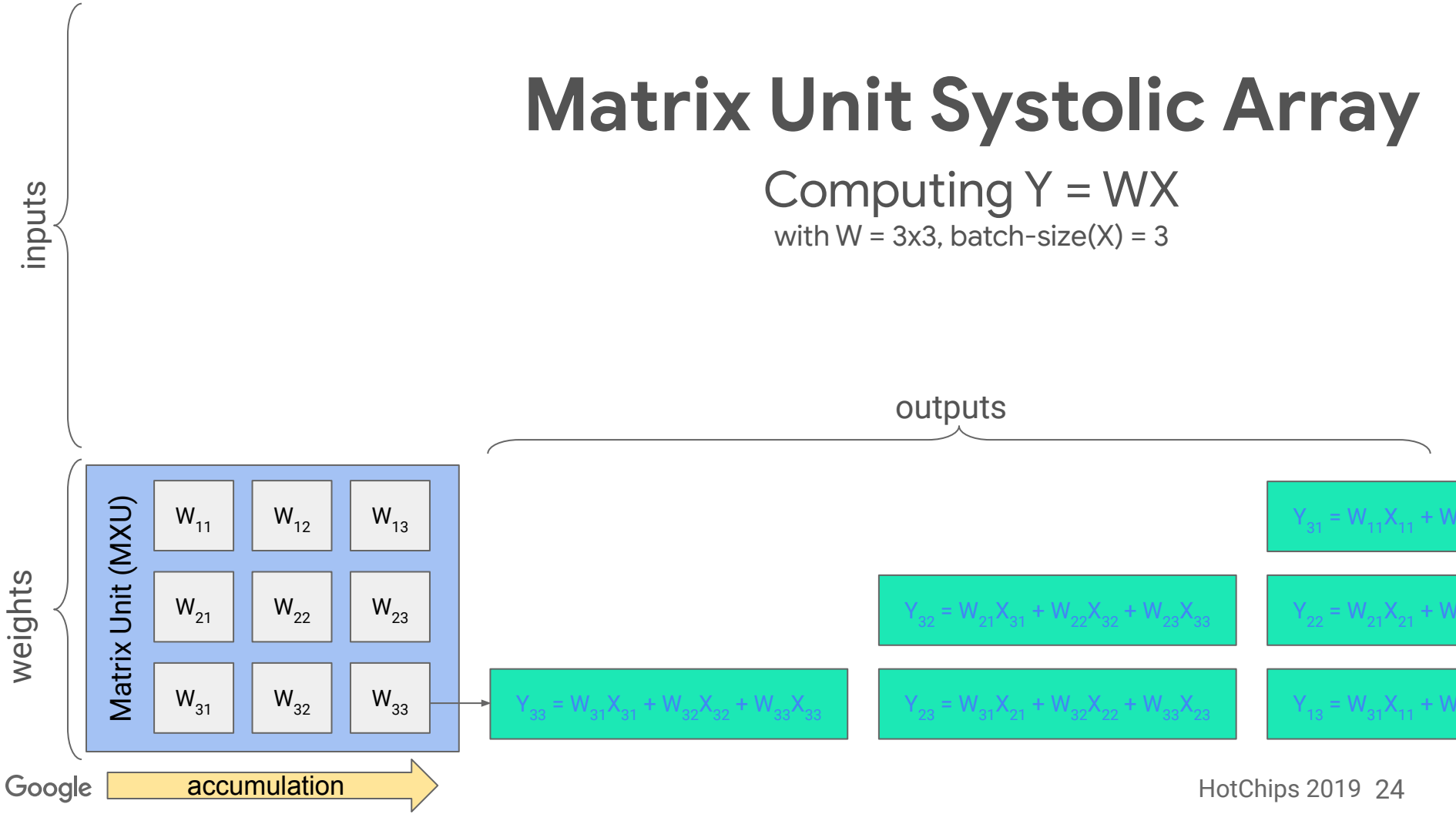
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

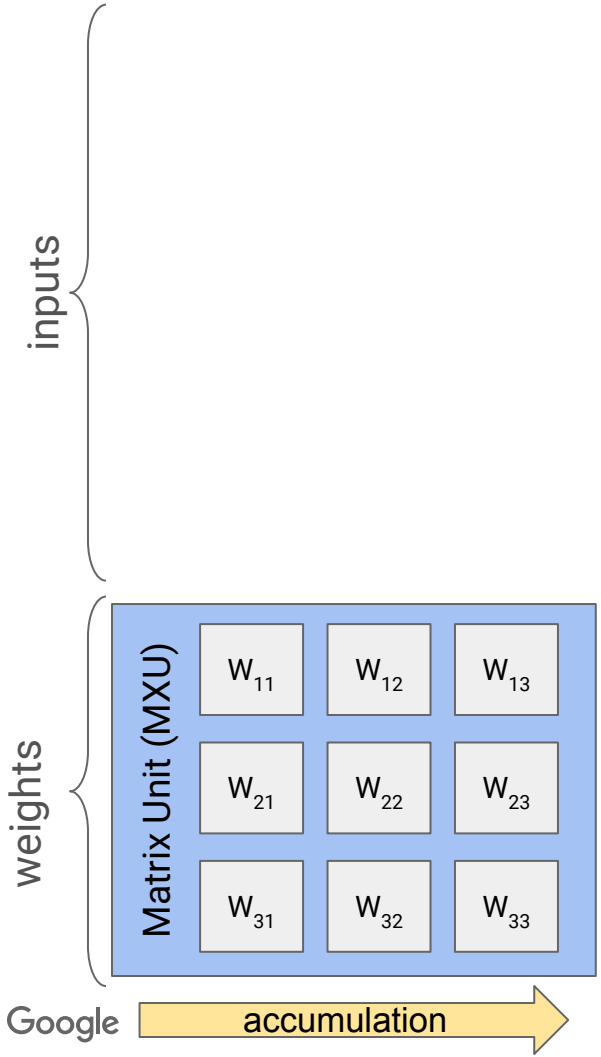
with W = 3x3, batch-size(X) = 3



Matrix Unit Systolic Array

Computing Y = WX

with W = 3x3, batch-size(X) = 3

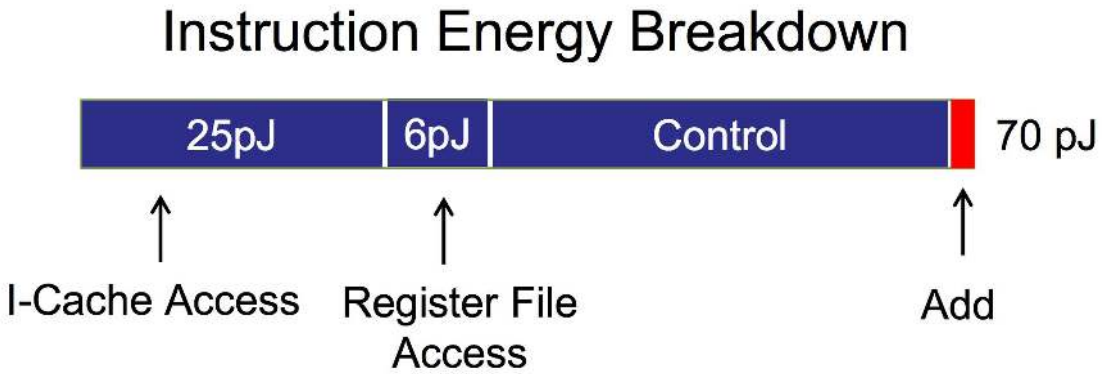


Very high throughput
for matrix multiplications

Systolic Arrays and Energy Efficiency

From Mark Horowitz's ISSCC 2014 Keynote: "Computing's Energy Problem (and what we can do about it)".

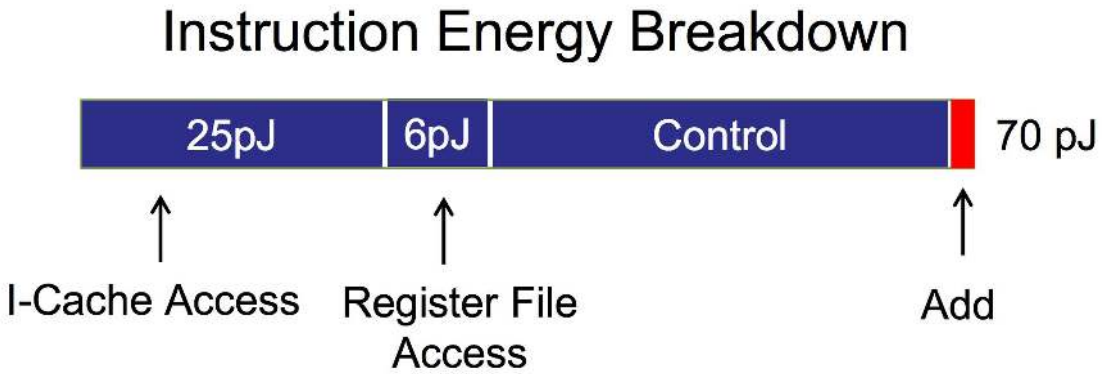
Energy for control logic, SRAM, and register accesses needed by matrix multiply dominates in conventional processors.



Systolic Arrays and Energy Efficiency

This doesn't even count DRAM energy cost, which is orders of magnitude larger (1300-2600pJ).

Now consider doing 16,384 operations per clock in a systolic array, which has high reuse so only the red portion scales up.



Systolic Arrays and Energy Efficiency

For this figure

Now consider

c array.

Each operand

Access

TPU Codesign

- **ML research**: latency and BW requirements.
- **Software**: controllability.
- **Systems**: thermal limits.

Numerics

Neural-network researchers showed that inference did not need float32 precision.

Vanhoucke, V., Senior, A. and Mao, M.Z., 2011, December. Improving the speed of neural networks on CPUs.
In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop* (Vol. 1, p. 4).

TPUv1 deployed with int8 hardware and support for int16 through software.

int16 intended as “insurance”; used mostly in LSTM-based models.

Training has traditionally been done in floating point.

Q: Does training require full float32 resolution?
A: Not everywhere. But where matters.

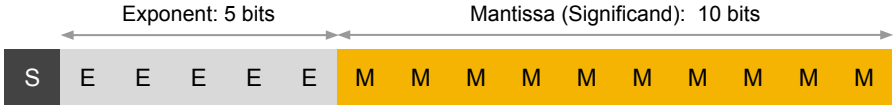
Numerics

float32: Single-precision IEEE Floating Point Format

Range: ~1e^{-38} to ~3e^{38}

float16: Half-precision IEEE Floating Point Format

Range: ~5.96e^{-8} to 65504



bfloat16: Brain Floating Point Format

Range: ~1e^{-38} to ~3e^{38}

Numerics

float32: Single

Expon

S E E E E

e^{-38} to ~$3e^{38}$

M M M M M

float16: Half-p

Exponent: 5 bits

S E E E E

$.96e^{-8}$ to 65504

bfloat16: Brai

Expon

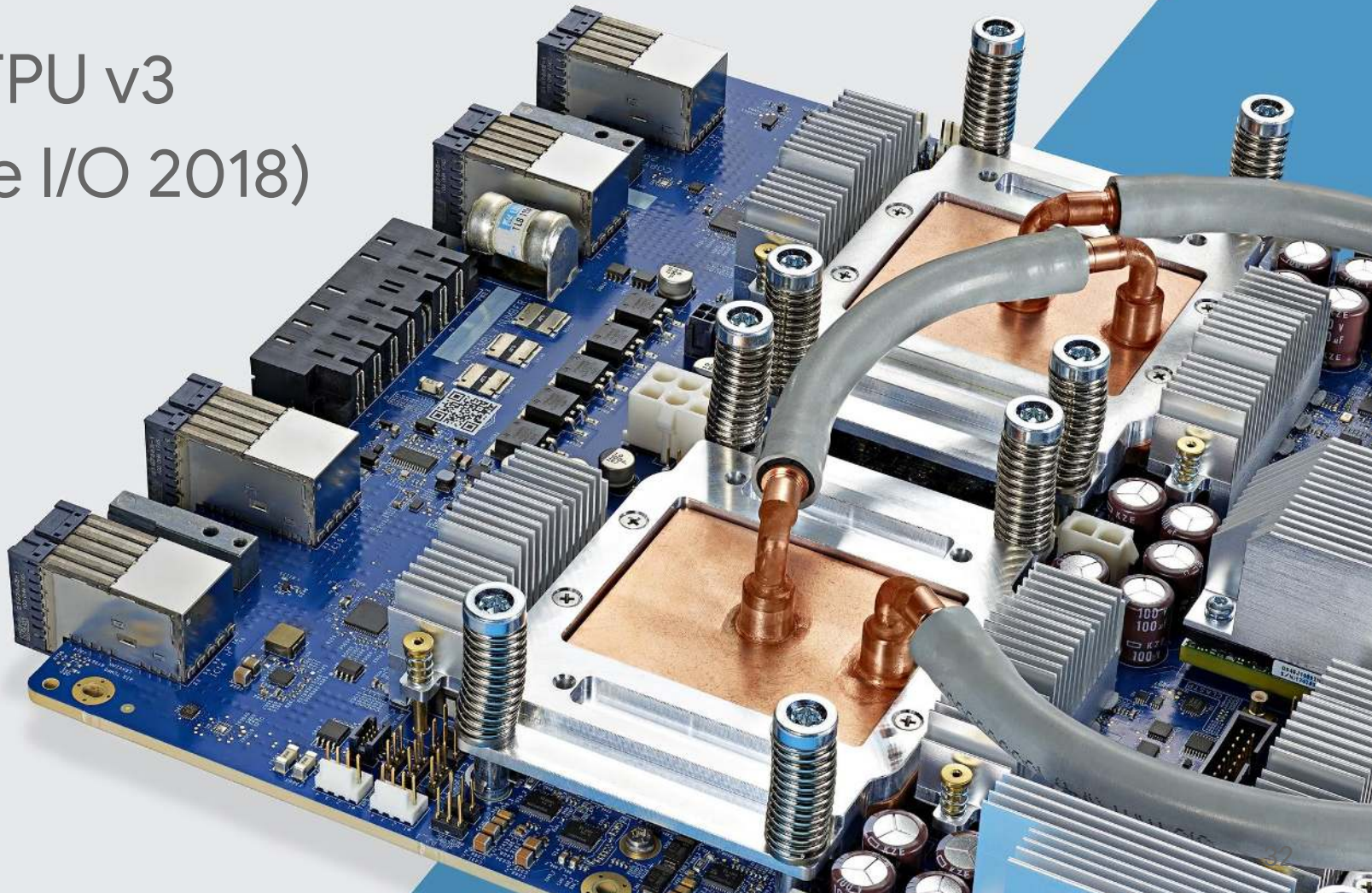
S E E E E

e^{-38} to ~$3e^{38}$

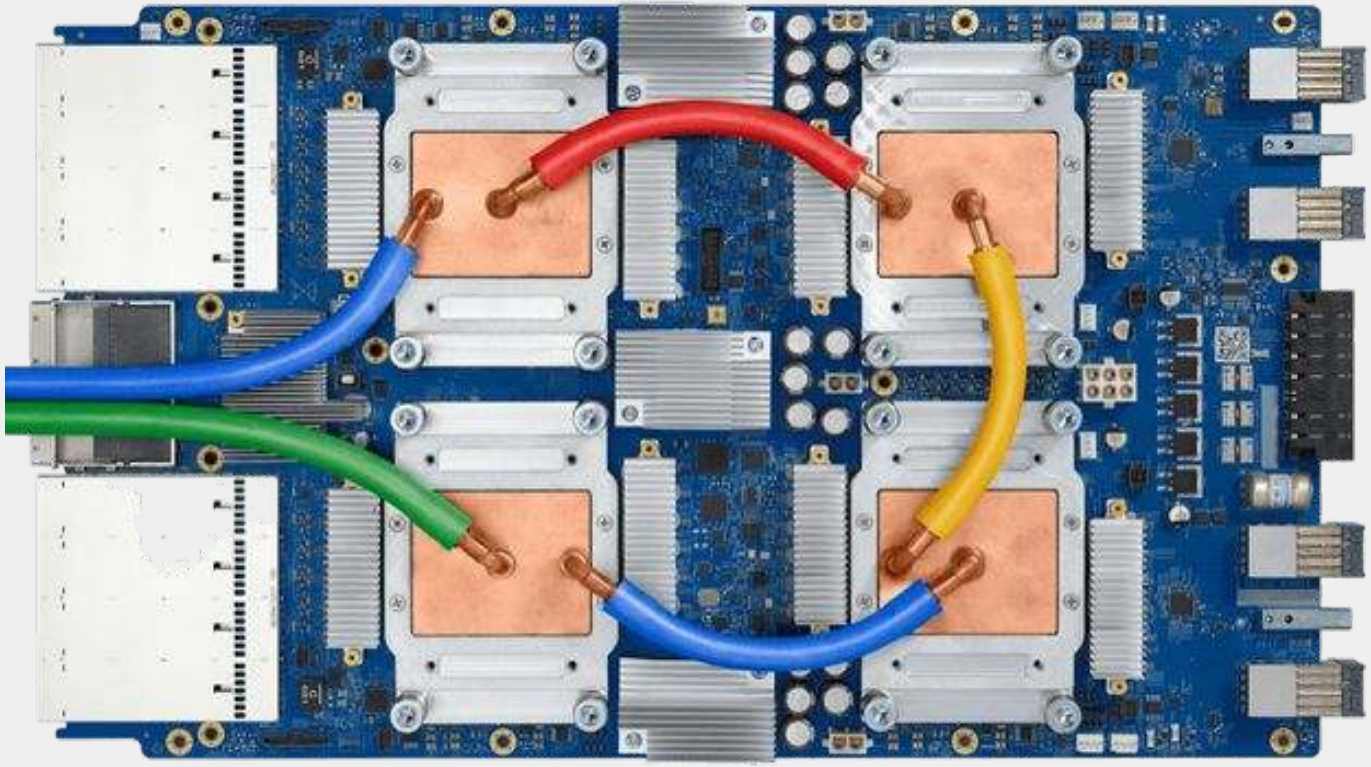
TPU Codesign

- **ML research**: trains without loss scaling, unlike float16 [Micikevicius 2017].
- **ASIC**: small mantissa reduces multiplier power, area.
- **Software**: same dynamic range as float32, same Inf/NaN behavior.

Cloud TPU v3
(Google I/O 2018)

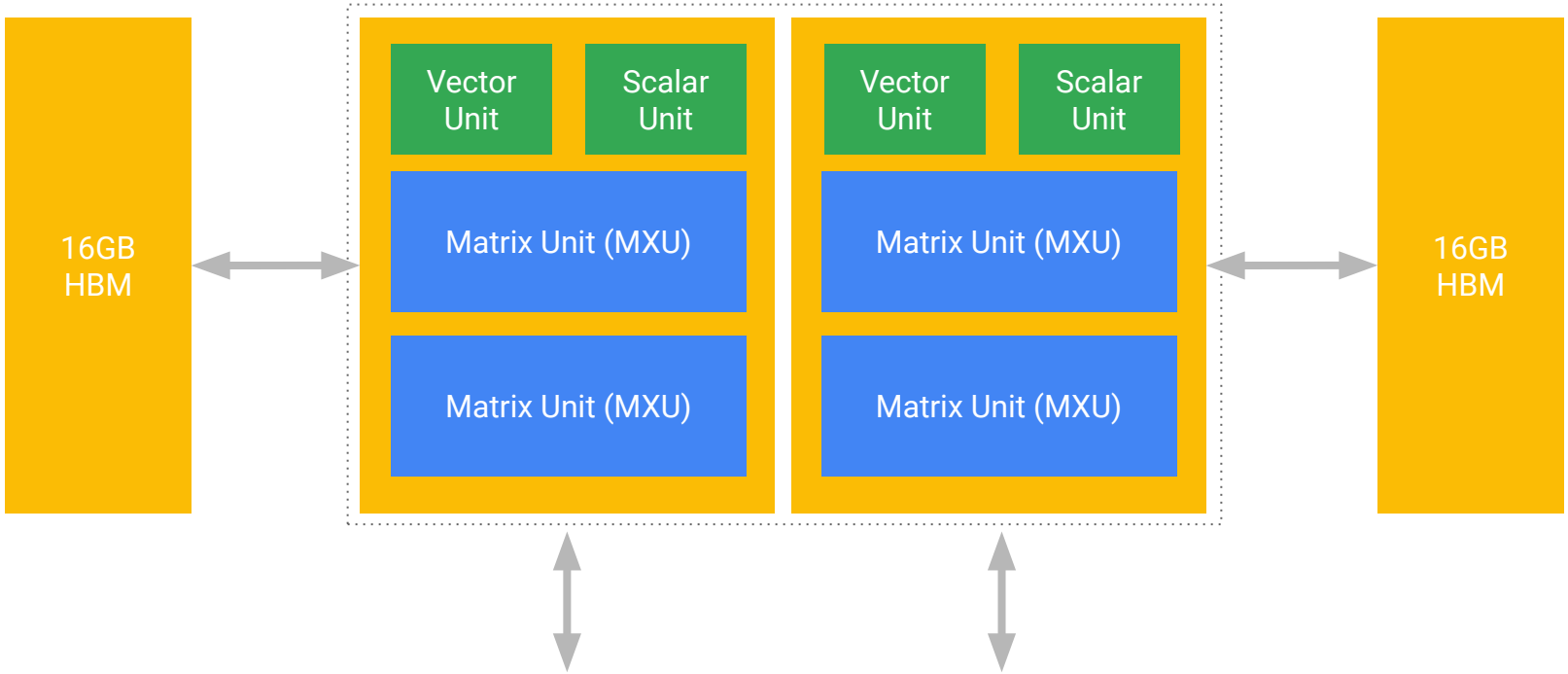


Cloud TPU v3 Board



- 420 teraflops of computation, 128 GB of HBM memory
- Liquid cooling

Cloud TPU v3 Chip Architecture



Cloud TPU v3 Chip Architecture

16GB HBM

16GB HBM

TPU Codesign

- **ML Research**: latest computational requirements.
- **Systems**: liquid cooling.
- **Data Center**: space provisioning, network requirements.

Datacenter-Scaled Tiled Architecture

Three ways to spend on computer architectural resources:

- Compute
- Memory
- **Custom interconnect**

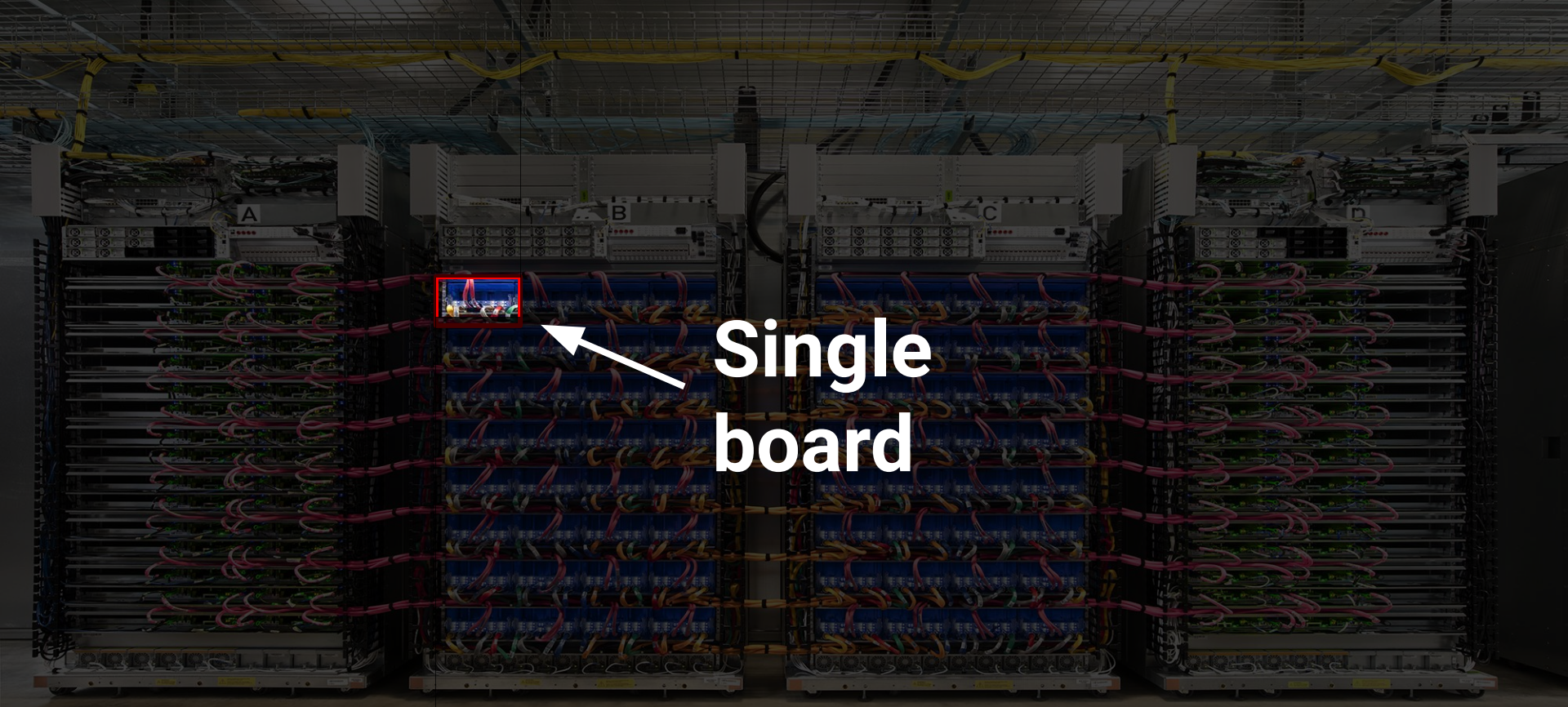
Allows us to scale the system by connecting many boards together into a super-computing pod to work on a single workload.

Datacenter-Scaled Tiled Architecture

First, about the custom interconnect itself:

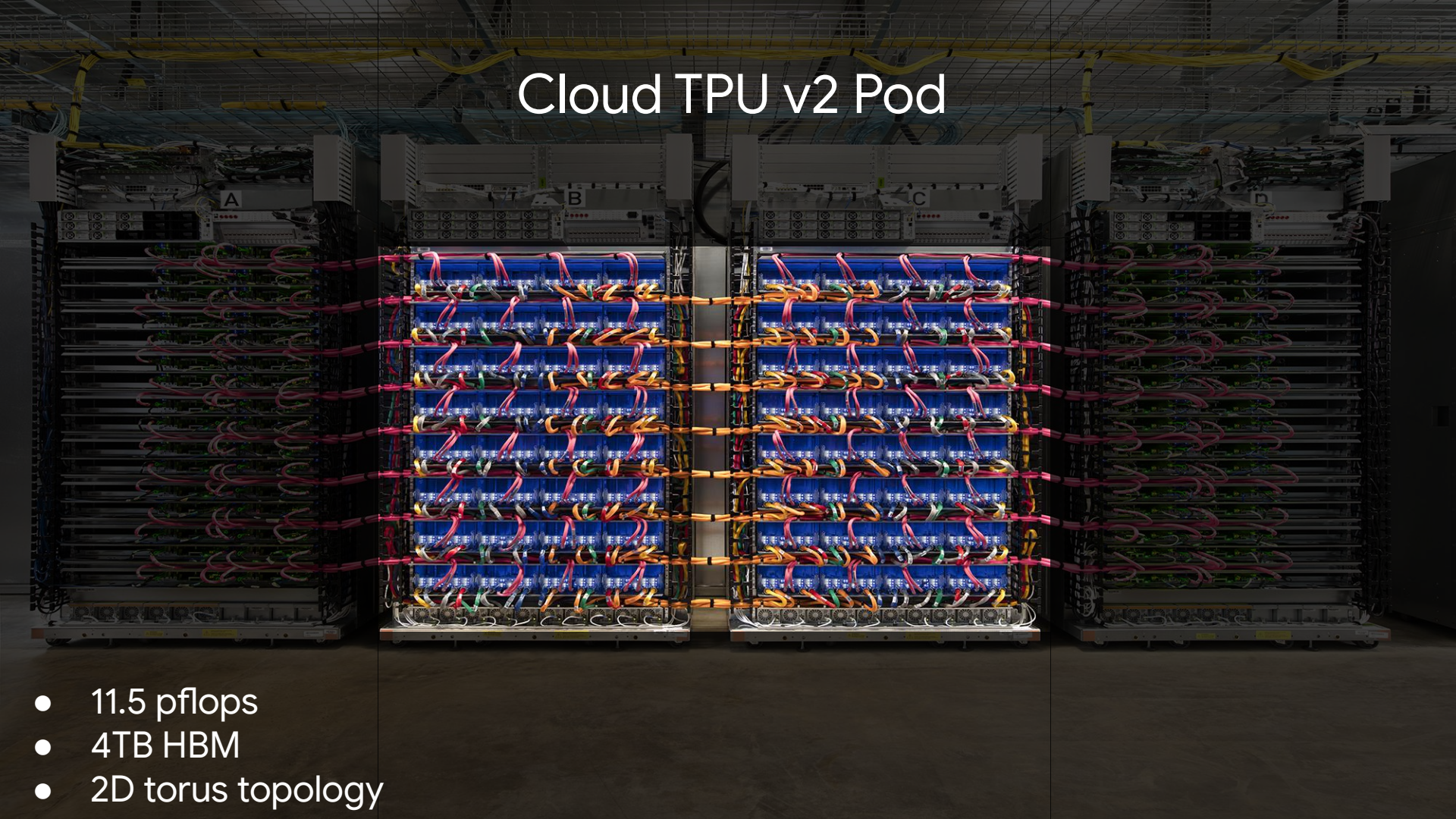
- 2-D torus network topology
- Custom protocol allows lower overhead than generic protocols (eg ROCE, Infiniband)
- Enables algorithms like parallel in-network reduction to a single node
- Integrated router into the TPU chip

Now, let’s use the interconnect to scale the system...



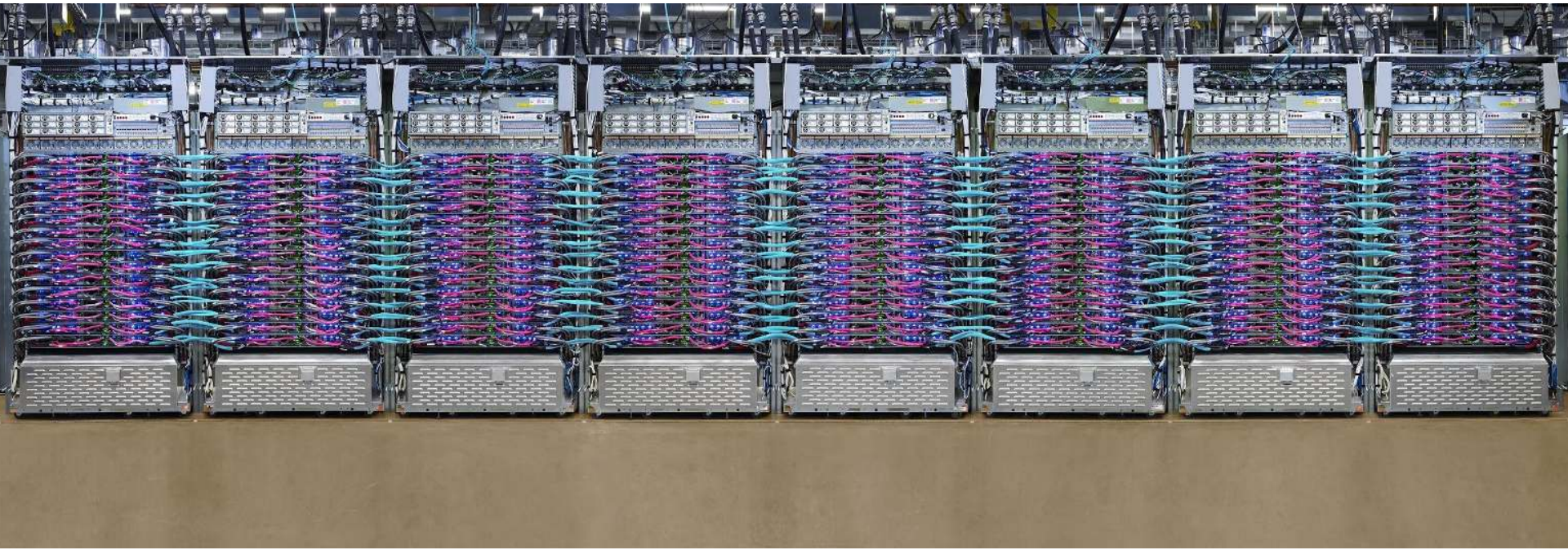
Cloud TPU v2 Board

Cloud TPU v2 Pod

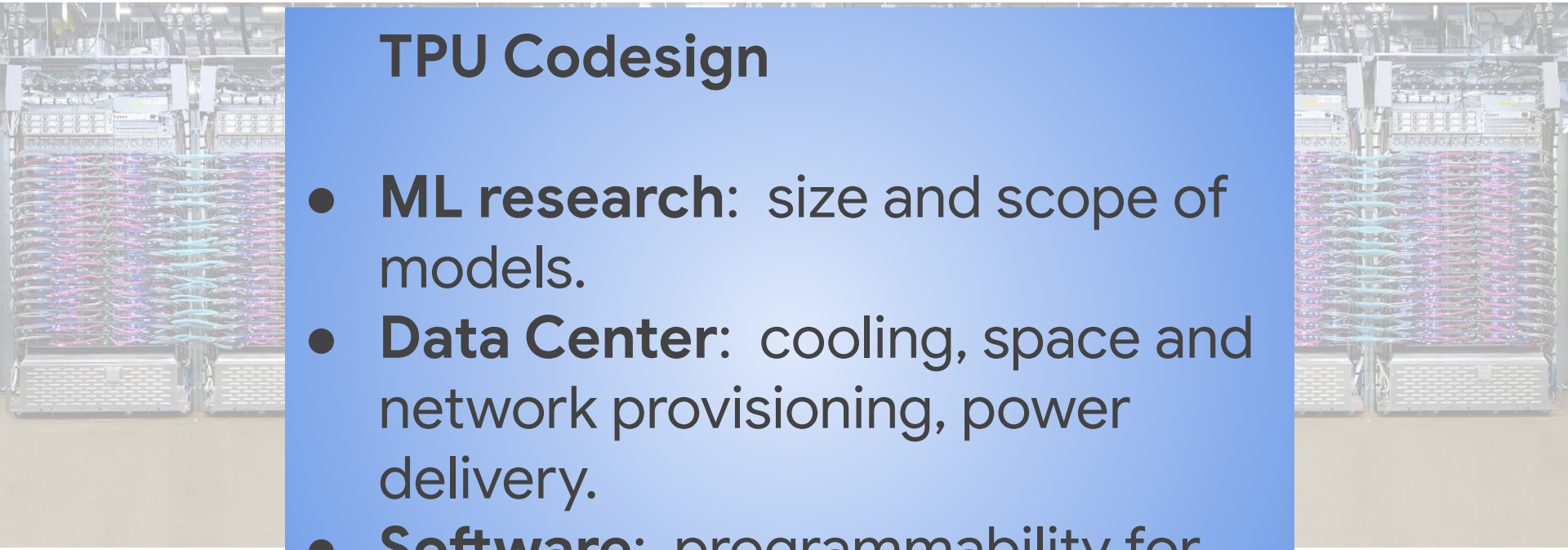


- 11.5 pflops
- 4TB HBM
- 2D torus topology

Cloud TPU v3 Pod



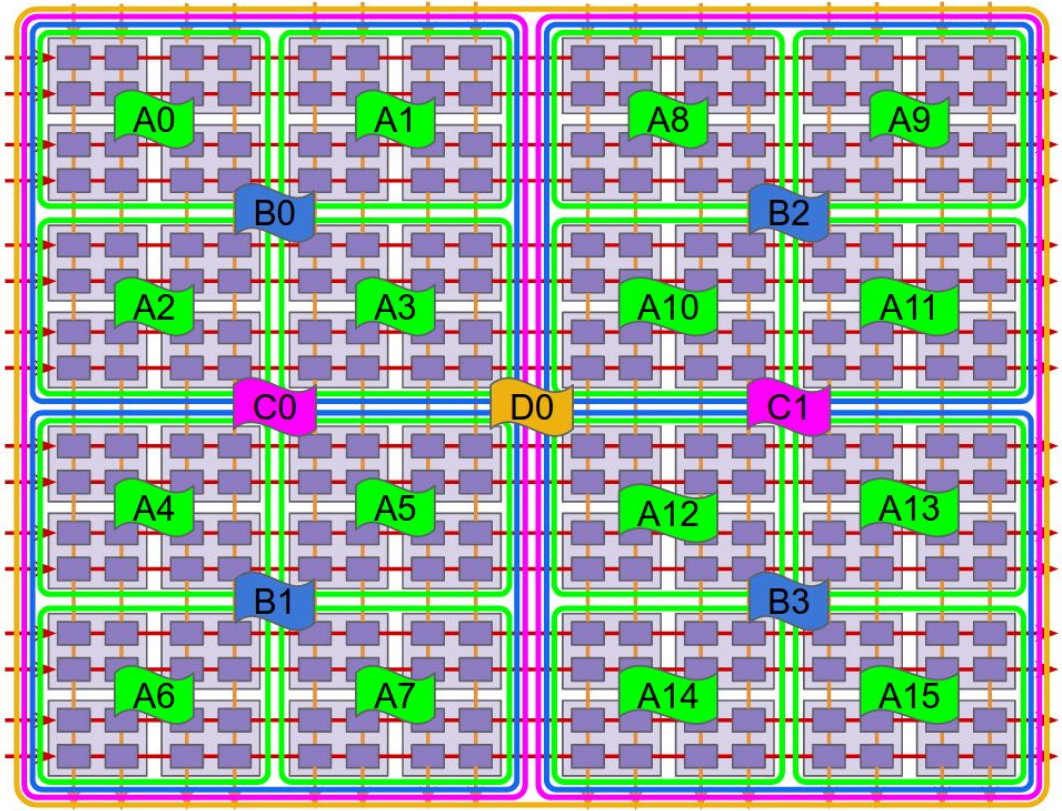
- >100 pflops
- 32TB HBM



TPU Codesign

- **ML research**: size and scope of models.
- **Data Center**: cooling, space and network provisioning, power delivery.
- **Software**: programmability for parallelism infrastructure.

Example Slicing of TPU Pod Hardware

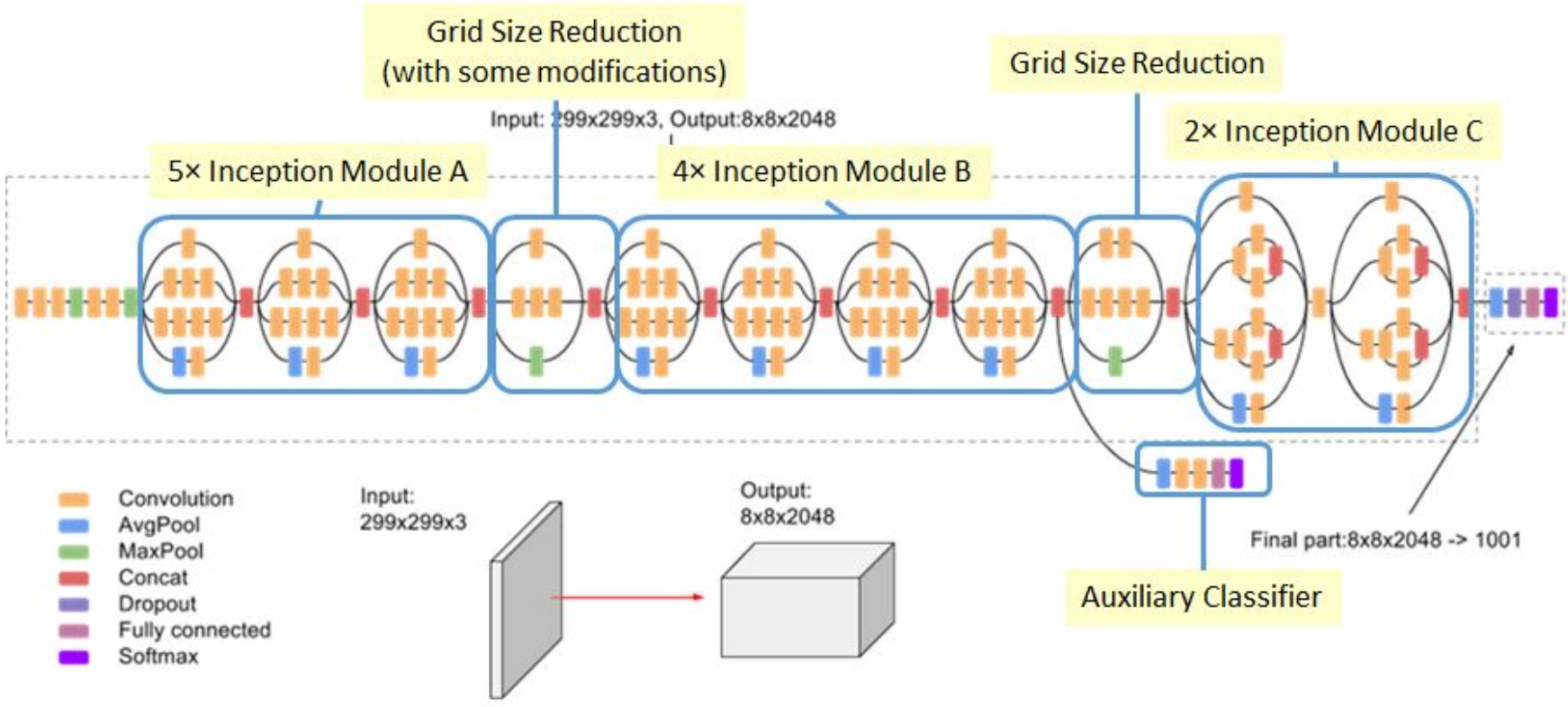


Cloud TPU Sizes

https://cloud.google.com/tpu/docs/types-zones

US | EUROPE | ASIA PACIFIC

TPU type (v2)	TPU v2 cores	Total TPU memory	Available zones
v2-8	8	64 GiB	europe-west4-a
v2-32 (Beta)	32	256 GiB	europe-west4-a
v2-128 (Beta)	128	1 TiB	europe-west4-a
v2-256 (Beta)	256	2 TiB	europe-west4-a
v2-512 (Beta)	512	4 TiB	europe-west4-a
TPU type (v3)	**TPU v3 cores**	**Total TPU memory**	**Available zones**
v3-8	8	128 GiB	europe-west4-a
v3-32 (Beta)	32	512 GiB	europe-west4-a
v3-64 (Beta)	64	1 TiB	europe-west4-a
v3-128 (Beta)	128	2 TiB	europe-west4-a
v3-256 (Beta)	256	4 TiB	europe-west4-a
v3-512 (Beta)	512	8 TiB	europe-west4-a
v3-1024 (Beta)	1024	16 TiB	europe-west4-a
v3-2048 (Beta)	2048	32 TiB	europe-west4-a


Grid Size Reduction
(with some modifications)
Input: 299x299x3, Output:8x8x2048
Grid Size Reduction
2× Inception Module C
5× Inception Module A
4× Inception Module B
Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax
Input:
299x299x3
Output:
8x8x2048
Final part:8x8x2048 -> 1001
Auxiliary Classifier

Actual Models Map to TPU Architecture

Matrix Unit

→ High-FLOPS Ops

- Matrix multiplication (dense layers)
- Convolutions

Vector Units

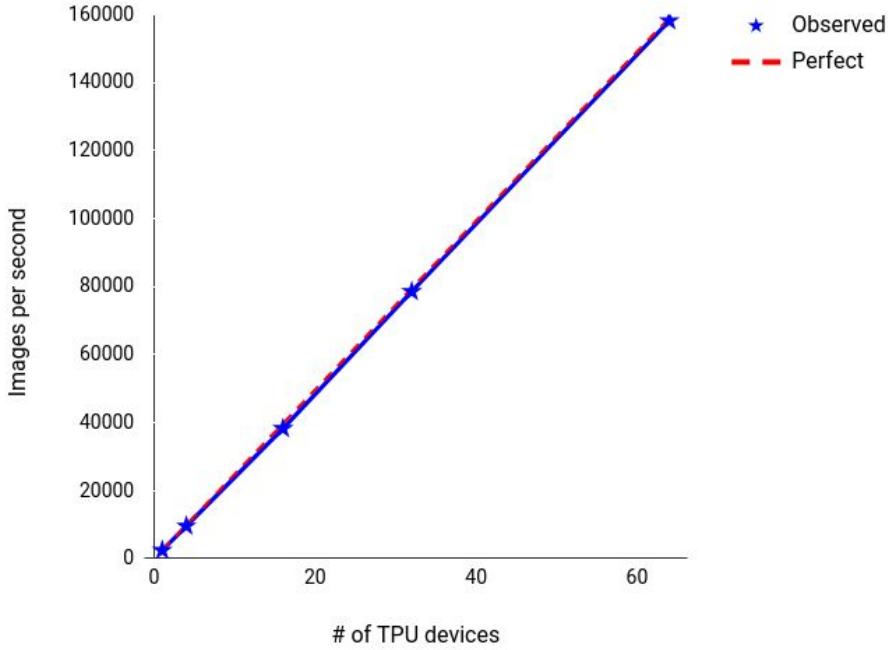
→ Element-wise Ops

- Non-linearities
- Activations
- Dropout
- Pooling
- Parameter gradient updates

Scalar Unit

- TPU control
- Data flow
- Memory addressing (concat's)

Linear Performance Scaling



```
import tf.keras.layers as l
import tf.losses.sparse_softmax_cross_entropy as softmax

def model_fn(images, labels, mode):
  conv = l.Conv2D(32, 5)(images)
  flat = l.Flatten()(conv)
  logits = l.Dense(10)(flat)
  loss = softmax(labels, logits)
  optimizer =
    t                                    zer(

  opt
    tf.contrib.tpu.CrossShardOptimizer(
        optimizer)
  train_op = optimizer.minimize(loss)
  return
    tf.contrib.tpu.TPUEstimatorSpec(
        loss=loss,
        train_op=train_op)
```

Runs on TPU

```
import tensorflow as tf
import tf.contrib.cluster_resolver.TPUClusterResolver

def input_fn(params):
  ds = tf.data.TFRecordDataset(path)
  ds
  ds
  ds                                     0)
  ds = ds.batch(params['batch_size'])
  return ds
```

Runs on CPU

```
cr = TPUClusterResolver()
rc = tf.contrib.tpu.RunConfig(
    cluster=cr,
    model_dir=FLAGS.model_dir)
est = tf.contrib.tpu.TPUEstimator(
    model_fn=model_fn, use_tpu=True,
    train_batch_size=1024)
est.train(input_fn=input_fn)
```

```
import tf.keras.layers as l
import tf.losses.sparse_softmax_cross_entropy as softmax
```

```
import tensorflow as tf
import tf.contrib.cluster_resolver.TPUClusterResolver
```

```
def model_fn(images, labels, mode):
  conv = l.Con
  flat = l.Fla
  logits = l.D
  loss = softm
  optimizer =
    t
  opt
    tf.contrib
        optimi
  train_op = o
  return
    tf.contrib
        loss=loss,
        train_op=train_op)
```

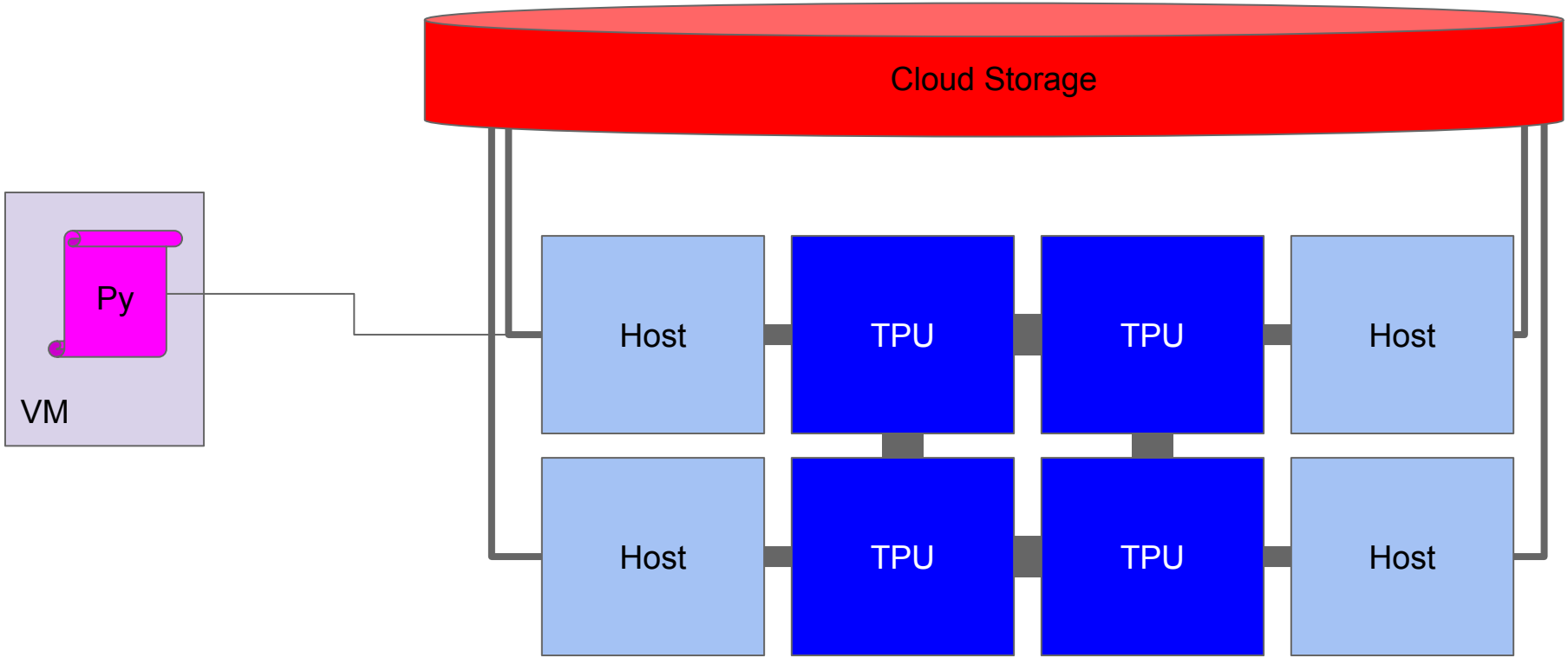
```
def input_fn(params):
                    t(path)
                    0)
                    _size'])
                    (
                    )
                    ator(
                    u=True,
    train_batch_size=1024)
est.train(input_fn=input_fn)
```

Run

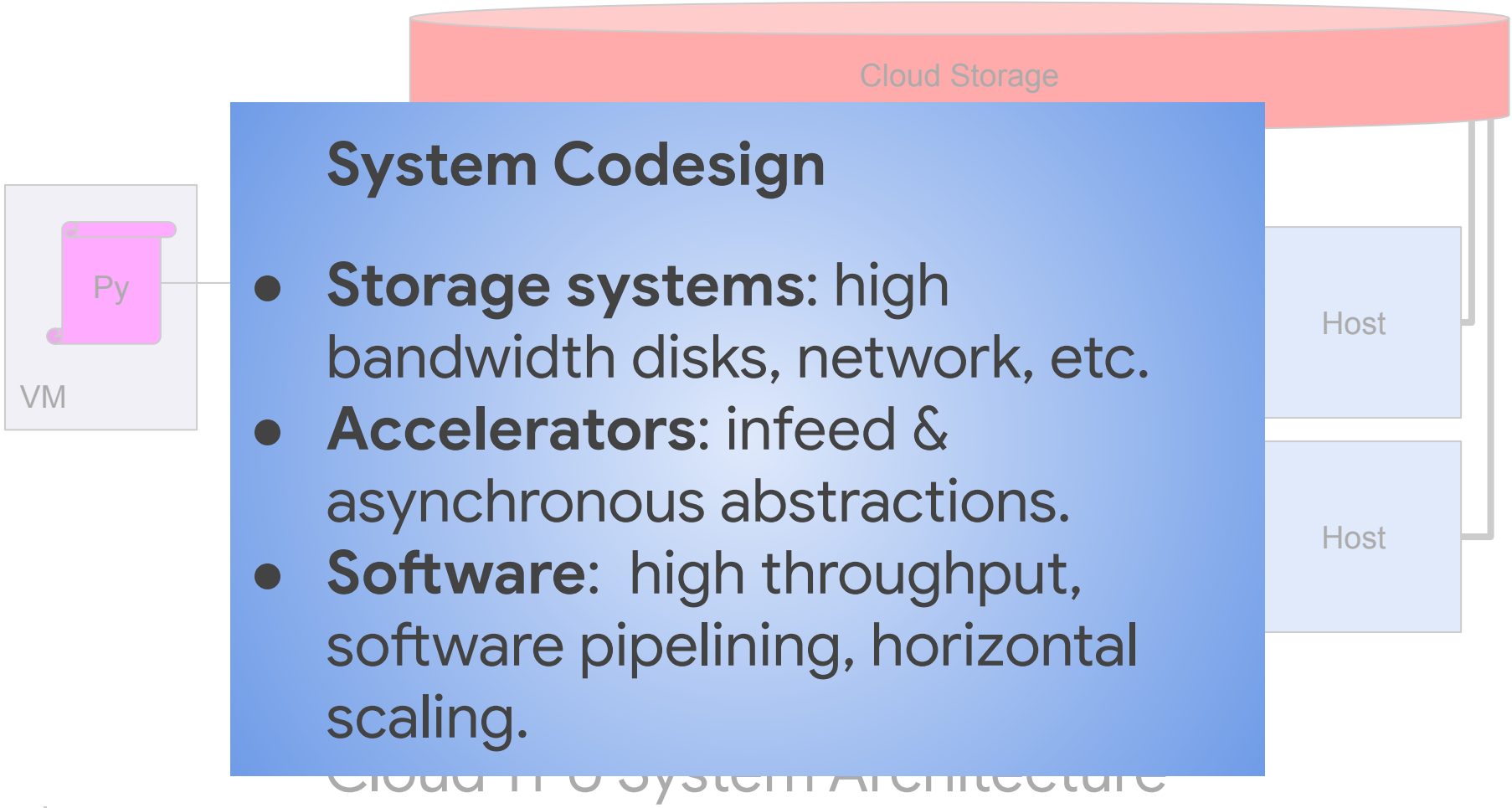
TPU

Zero code changes when scaling from a single device to a whole pod!

- **ML research**: influence abstractions.
- **Compilers**: JIT code-gen & parallel IRs.
- **APIs**: Carefully constructed for distribution.



Cloud TPU System Architecture



System Codesign

- **Storage systems**: high bandwidth disks, network, etc.
- **Accelerators**: infeed & asynchronous abstractions.
- **Software**: high throughput, software pipelining, horizontal scaling.

Cloud TPU System Architecture

Streaming Data to Cloud TPUs at High Speed

Cloud Storage

A distributed file system optimized for extreme scale and low cost.

Compute Engine

Stream data from Compute Engine VM(s) local file system or memory for maximum flexibility.

Cloud Bigtable

Scalable, low-latency wide-row structured storage system.

Results

MLPerf 0.6: TopLine* Comparison (mlperf.org)

E2E time (units in secs; # of chips in parentheses)

	TPUv3		Tesla V100		Speedup
Resnet-50	(1024)	77	(1536)	80	4%
SSD	(1024)	72	(480)	134	86%
Mask-RCNN	(128)	2136	(192)	1107	-48%
GNMT	(512)	127	(384)	108	-15%
Transformer	(1024)	51	(480)	95	86%

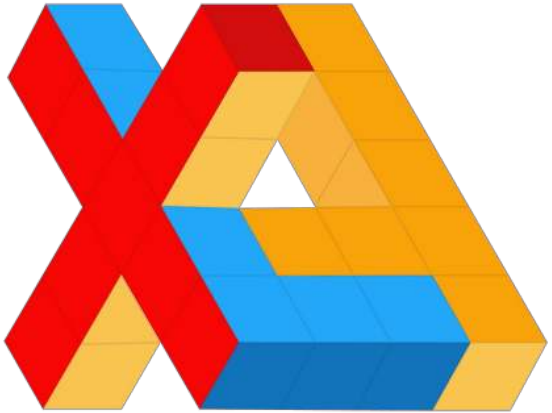
* TopLine: fastest convergence time (regardless of resource consumption)

Performance & Optimization Tips

Performance & Optimization Tips

On-device execution

- Leverage MXU size
- Avoid reshapes (XLA helps a lot!)
- Scale out for trivial data-parallel improvements (interconnect awesomeness!)
- More advanced parallelism available



ACCELERATED LINEAR ALGEBRA

Performance & Optimization Tips

Input pipelines

- Many parallel, large, streaming reads
- Software pipelining critical
- Tradeoffs between storage, bandwidth, and compute to tune input pipeline
  - Leverage CPU parallelism

Use the profiler!

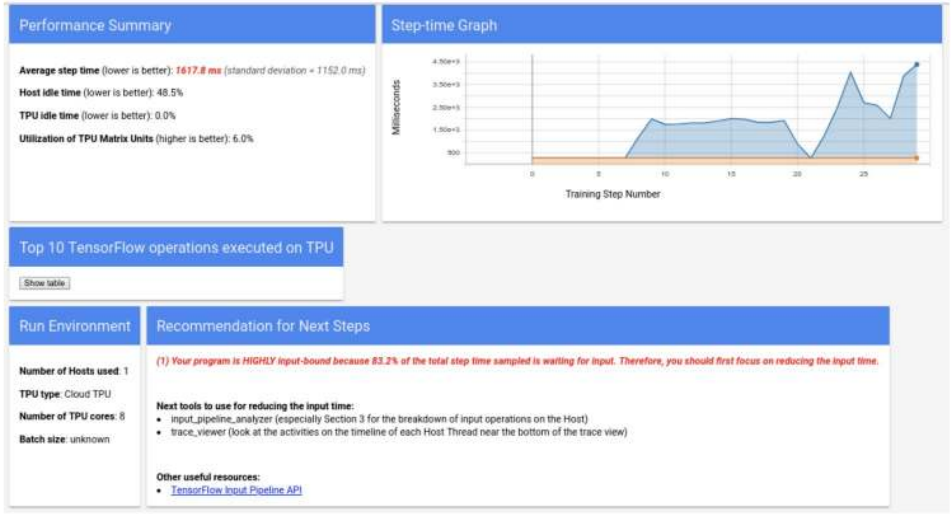
Performance & Optimization Tips

From: https://cloud.google.com/tpu/docs/cloud-tpu-tools#profile_tab

Profile overview page

The overview page (`overview_page`), available under **Profile**, provides a top level view of how your model performed during a capture run. The page shows you an aggregated overview page for all the TPUs, as well as an overall input pipeline analysis. There is an option for selecting individual TPUs in the Host dropdown.

The page displays data in the following panels:



Codesign Summary

1. Necessary due to EoML
2. Architect infrastructure for usability and scale!
3. Leverage all areas of expertise

We already co-design:

1. Data centers
2. Power & cooling systems
3. Board topology
4. On-chip compute units
5. Accelerator-accelerator networks
6. Numerics
7. Memory & cache hierarchies
8. Compilers
9. ML frameworks & APIs
10. Datacenter networks
11. Data storage hardware & software
12. File formats
13. Accelerator-host interfaces & bandwidth
14. Cluster scheduling systems
15. ML Algorithms
16. Model partitioning & parallelism research
17. Security

What else can we co-design?

Co-design the *programming language* too

Check out **Swift for TensorFlow** at:

https://github.com/tensorflow/swift

Please join our open design reviews on Friday's!
Sign up at: swift@tensorflow.org (Google Group)

Thank you!